The Finite Difference Method

Charles Collicutt

March 30, 2012

Discretization

Suppose we wish to numerically model some physical phenomenon in a certain volume of space.

The first step will be to discretize that continuous space into a grid of discrete cells. We will then be able to represent physical quantities within that space as numbers associated with each cell.

Many physical phenomena can be modelled by differential equations. We will therefore need a way to numerically approximate the solutions to differential equations using our discretized grid.

Approximating Derivatives

The derivative of $f(x)$ is defined like so:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

If we are to approximate a derivative numerically then we cannot actually have an h of zero. h corresponds to the spacing of our grid, the granularity of our discretization, the width of each cell. It is sometimes known as the *step size*. If h were zero then we would have a continuous space again.

We need to approximate $f'(x)$ where h is a fixed value.

Taylor's Theorem

Taylor's theorem says that, assuming $f(x)$ is continuously differentiable n times,

$$f(x+h) = f(x) + \frac{f'(x)}{1!}h + \frac{f''(x)}{2!}h^2 + \ldots + \frac{f^n(x)}{n!}h^n + R_n(x+h)$$

Where $R_n(x+h)$ is a remainder term denoting the difference between the Taylor polynomial of degree n and the actual value of $f(x+h)$.

This remainder term can be expressed in various ways, one of which is the Lagrange form. Here it is stated that there exists a number ξ between x and $x+h$ such that

$$R_n(x+h) = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$$

An Approximation

Consider the first order Taylor polynomial:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(\xi)}{2}h^2$$

Rearrange it:

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{f''(\xi)}{2}h$$

Note that on the right-hand side of the equality we have the function that we wish to approximate, $f'(x)$, plus a remainder term.

Therefore the term on the left-hand side of the equality is an approximation to $f'(x)$ with an error proportional to h. We call this error the *truncation error*.

Intuitive Derivation

Recall the definition of the derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

It makes intuitive sense that this

$$\frac{f(x+h) - f(x)}{h}$$

should be an approximation to $f'(x)$ whose error decreases as h gets smaller.

The derivation of the approximation from the Taylor polynomial proves that the error is $O(h)$.

Finite Differences

A finite difference is any mathematical expression of the form

$$f(x+a) - f(x-b)$$

The numerator of our approximation from the last slide is known as the *Forward Difference*:

$$\Delta_h[f](x) = f(x+h) - f(x)$$

Two other commonly used finite differences are the *Backward Difference*:

$$\nabla_h[f](x) = f(x) - f(x-h)$$

And the *Central Difference*:

$$\delta_h[f](x) = f(x+\frac{h}{2}) - f(x-\frac{h}{2})$$

Backward Difference

Recall the Taylor polynomial for $f(x-h)$:

$$f(x-h) = f(x) - f'(x)h + \frac{f''(\xi)}{2}h^2$$

Rearrange to obtain this:

$$\frac{f(x) - f(x-h)}{h} = f'(x) + \frac{f''(\xi)}{2}h$$

And we see that if we approximate $f'(x)$ using the backward difference we get a truncation error of $O(h)$ just as we did with the forward difference.

Central Difference

Consider the Taylor polynomial for $f(x+\frac{h}{2})$:

$$f(x+\frac{h}{2}) = f(x) + \frac{f'(x)h}{2} + \frac{f''(x)h^2}{8} + \frac{f'''(\xi)}{48}h^3$$

And for $f(x-\frac{h}{2})$:

$$f(x-\frac{h}{2}) = f(x) - \frac{f'(x)h}{2} + \frac{f''(x)h^2}{8} + \frac{f'''(\xi)}{48}h^3$$

Therefore:

$$\frac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{h} = f'(x) + \frac{(f'''(\xi_1) - f'''(\xi_2))h^2}{48}$$

The truncation error is $O(h^2)$. As h gets smaller, the error will tend to zero faster than it would if it were $O(h)$. Thus this is a better approximation than the forward or backward difference.

Fractional Step Sizes

Note that the central difference as presented on the last slide may be problematic. We don't really want to use fractions of h as that amounts to changing the step size.

We can achieve an approximation with a truncation error of the same order by averaging the forward and backward differences:

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + \frac{(f'''(\xi_1)-f'''(\xi_2))h^2}{6}$$

Orders of Accuracy

All the finite difference formulae introduced so far have been approximations for the same function: $f'(x)$

But they have different truncation errors:

$$\frac{f(x) - f(x-h)}{h} \Rightarrow O(h) \qquad \text{“First Order”}$$

$$\frac{f(x+h) - f(x)}{h} \Rightarrow O(h) \qquad \text{“First Order”}$$

$$\frac{f(x+h) - f(x-h)}{2h} \Rightarrow O(h^2) \qquad \text{“Second Order”}$$

We can construct formulae that approximate derivatives to an arbitrary order of accuracy.

Summing Subformulae

Let a_k be the coefficient by which we multiply $f(x + kh)$. Then:

$$\frac{f(x+h) - f(x-h)}{2h} = \frac{1}{h} \sum_{k=-1}^{1} a_k f(x + kh)$$

where

$$a_{-1} = -\frac{1}{2}$$
$$a_0 = 0$$
$$a_1 = \frac{1}{2}$$

We can represent any finite difference formula as a sum of subformulae in this way.

With a bit of linear algebra we can ensure that, when the subformulae are summed, the unwanted terms in the Taylor series sum to zero.

Choosing Coefficients (1)

Represent the k terms in the Taylor polynomial for $f(x+kh)$ as a column vector. Let A be a matrix composed of such vectors for various k.

There will be one column for each k, i.e. each offset from x, that we wish to consider.

The first row will correspond to the $f(x)$ term in each polynomial, each further row will correspond to a derivative from $f'(x)$ up to $f^{(n)}(x)$ where $n+1$ is the order of accuracy we wish to achieve.

Choosing Coefficients (2)

To obtain appropriate coefficients such that when we sum the polynomials we are left with only $f'(x)$, just solve the following equation for c:

$$Ac = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

For example, the central difference can be derived like this:

$$\begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} -\frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Examples of Higher Orders of Accuracy

Here are some central finite difference approximations to $f'(x)$ of various orders of accuracy:

Accuracy	a_{-4}	a_{-3}	a_{-2}	a_{-1}	a_0	a_1	a_2	a_3	a_4
$O(h^2)$				$-\frac{1}{2}$	0	$\frac{1}{2}$			
$O(h^4)$			$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$		
$O(h^6)$		$-\frac{1}{60}$	$\frac{3}{20}$	$-\frac{3}{4}$	0	$\frac{3}{4}$	$-\frac{3}{20}$	$\frac{1}{60}$	
$O(h^8)$	$\frac{1}{280}$	$-\frac{4}{105}$	$\frac{1}{5}$	$-\frac{4}{5}$	0	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$

Other Derivatives

So far we have only constructed approximations to $f'(x)$. We could use the same technique to construct approximations to other derivatives.

For example, if we wish to approximate $f''(x)$ rather than $f'(x)$, then instead of solving this for c:

$$Ac = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \text{We solve this:} \qquad Ac = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

By changing the vector on the right-hand side of the equation, we can construct an approximation to any linear combination of derivatives.

Round-off Error

Unfortunately, truncation of the Taylor series is not the only source of error. We must also contend with *round-off error*.

In a practical computing system we have only a limited number of bits available to represent each number, so we cannot represent all real numbers exactly but must round them off to the nearest representable number.

This is analogous to an attempt to represent $\frac{1}{3}$ in decimal notation with only a finite number of digits.

Calculating Error

Let

$$f(x) = g(x) + e(x)$$

where $e(x)$ is the round-off error when computing $f(x)$. Then

$$f'(x) = \frac{g(x+h) - g(x-h)}{2h} + E(f,h)$$

where $E(f,h)$ is the total error including both truncation error and round-off error:

$$E(f,h) = \frac{e(x+h) - e(x-h)}{2h} + \frac{(f'''(\xi_1) - f'''(\xi_2))h^2}{6}$$

Assume that $|e(x)| < \epsilon$ and $|(f'''(\xi_1) - f'''(\xi_2))| < M$:

$$|E(f,h)| < \frac{\epsilon}{h} + \frac{Mh^2}{6}$$

The Step-Size Dilemma

Note that the round-off error is inversely proportional to h but the truncation error is proportional to h^2. Therefore:

- If we decrease h in order to decrease the truncation error, we increase the round-off error.
- If we increase h in order to decrease the round-off error, we increase the truncation error.

Note that if we know ϵ and M we can calculate the optimal h. We typically do know ϵ as it is a property of whichever computing system we choose to use.

We might know M, especially if we are approximating a trigonometric function and therefore know, for example, that $\left|f^{(n)}(x)\right| \leq 1$.

Iterations

Usually, we are attempting to model some physical phenomenon as it changes over time.

So we don’t usually calculate finite differences only once: we calculate them repeatedly, each calculation giving us the state of the physical system in the next time step.

Note that this means we are discretizing time as well as space.

Numerical Stability

Given that we have an error in every calculation (due to both round-off and truncation), what happens to that error over repeated iterations?

If the difference between our approximation and the true solution remains constant or decreases, we say our method is numerically stable. If the error grows with each iteration then we say that it is unstable.

Different differential equations have different sensitivities to error, especially truncation error. Very sensitive equations, which are unstable unless h is very small, are known as *stiff* equations.

Numerical stability is a problem for all numerical methods, not just finite difference. Different methods differ in their ability to cope with stiff equations.

Implicit or Explicit

Suppose we know the state of a physical system at time t and we wish to calculate the state at time $t + 1$. There are two possible ways that the relationship between S_t and S_{t+1} could be expressed using differential equations, and hence two different styles of finite difference.

If we can calculate the next state directly from the current state like this:

$$S_{t+1} = f(S_t)$$

Then we call our method *explicit* finite difference.

If the next and current states are indirectly related like this:

$$f(S_t, S_{t+1}) = R$$

Then we call our method *implicit* finite difference.

Explicit Finite Difference

$$S_{t+1} = f(S_t)$$

Explicit finite difference calculations are typically quite fast, simply because we directly calculate a function of the current state in order to get the next state.

However, explicit finite difference is not very numerically stable. If the function involves stiff equations, we have to use a small h to keep the method stable, which may cancel out any benefit we receive from being able to calculate the next state directly.

If the differential equations are not very stiff then explicit finite difference will be one of the better ways to approximate their solution.

Implicit Finite Difference

$$f(S_t, S_{t+1}) = R$$

At every iteration of an implicit finite difference method, we must solve a system of equations. This typically involves a lot more computation than the direct computation of explicit finite difference.

In practice, it probably involves a sparse matrix operation, which is even worse on today's computing platforms.

However, implicit finite difference methods are much more numerically stable than explicit ones. Thus they can solve stiffer equations with larger step sizes.